The first step of any data related task is to inspect the data we are dealing with. This is crucial for data wrangling as well, since we need to explore the current structure of the data, in order to identify the required transformations.

- Inspect tabular data interactively with `View()`
- Examine the data structure of each object using `str()`

```
View(___)
str(___)
```

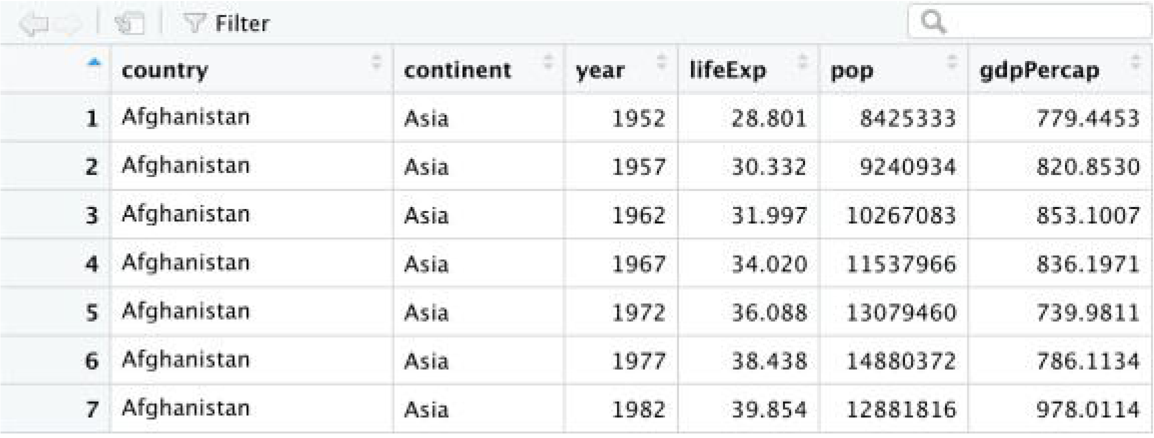
Interactive Inspection with View()

Before starting with any kind of data analysis, it is crucial to understand the data we are dealing with. Plotting is a very important tool to get a quick overview of the statistical properties of data and to detect possible outliers. However, visualization might not always be possible, due to the size or complexity of the data set.

As an alternative solution, it might be convenient to interactively dig through the data set. This could be done by a spreadsheet-like interface, similar to Microsoft Excel, which enables to filter, sort and inspect tabular data structures.

R provides the function `View()`, which shows an interactive data viewer. Depending on the used platform and editor, this viewer might look differently. Below you can see an example of the `View()` function in RStudio:

```
View(gapminder)
```



	country	continent	year	lifeExp	pop	gdpPercap
1	Afghanistan	Asia	1952	28.801	8425333	779.4453
2	Afghanistan	Asia	1957	30.332	9240934	820.8530
3	Afghanistan	Asia	1962	31.997	10267083	853.1007
4	Afghanistan	Asia	1967	34.020	11537966	836.1971
5	Afghanistan	Asia	1972	36.088	13079460	739.9811
6	Afghanistan	Asia	1977	38.438	14880372	786.1134
7	Afghanistan	Asia	1982	39.854	12881816	978.0114

Quiz: Interactive Inspection with View()

Why should you inspect data sets with `View()` before starting with your analysis?

- Get a first impression of the data quality.
- Find outliers and missing values.
- Interactively inspect the data set.
- Create reproducible outputs for reports.

Start Quiz

Exercise: Interactive Inspection with View()

Use the `View()` function on the `gapminder` data set and determine the country with the highest life expectancy. Pay also attention to year the projection was made. Set the variables `country` and `year` accordingly!

Start Exercise

Examining Data Structures with str()

Sometimes we need to analyze very large and complex data structures. Displaying these data sources may already be overwhelming and simply not possible with interactive tools. In these cases, the `str()` function comes to the rescue and prints the structure, as well as the first few values of any R object. Even very large and complex data structures can easily be displayed in the console that way.

As an example, let's take a look at structure of the `TitanicSurvival` data set:

```
library(carData)
str(TitanicSurvival)
'data.frame':   1309 obs. of  4 variables:
 $ survived      : Factor w/ 2 levels "no","yes": 2 2 1 1 1 2 2 1 2 1
...
 $ sex           : Factor w/ 2 levels "female","male": 1 2 1 2 1 2 1 2
1 2 ...
 $ age           : num  29 0.917 2 30 25 ...
 $ passengerClass: Factor w/ 3 levels "1st","2nd","3rd": 1 1 1 1 1 1 1
1 1 1 ...
```

It consists of three factor columns (`survived`, `sex` and `passengerClass`) and one numeric column `age`. Note, that for factor columns both the labels (e.g. `"no","yes"`) as well as the integer values are displayed.

Quiz: Examining Data Structures with str()

In which cases is it benefitial to use the `str()` function?

- Get an overview of highly complex data sets.
- Create summary statistics describing the data set.
- Plot histograms.
- Only for `data.frames`. `str()` can only handle `data.frames` and cannot be used for other objects.

Start Quiz

Quiz: Interpret the Output of str()

```
library(babynames)
str(babynames)
tibble [1,924,665 × 5] (S3: tbl_df/tbl/data.frame)
 $ year: num [1:1924665] 1880 1880 1880 1880 1880 1880 1880 1880 1880
1880 ...
 $ sex : chr [1:1924665] "F" "F" "F" "F" ...
 $ name: chr [1:1924665] "Mary" "Anna" "Emma" "Elizabeth" ...
 $ n   : int [1:1924665] 7065 2604 2003 1939 1746 1578 1472 1414 1320
1288 ...
```

```
 $ prop: num [1:1924665] 0.0724 0.0267 0.0205 0.0199 0.0179 ...
```

Examine the output of the `str()` function with the **babynames** dataset above. Which statements about the data set are correct?

- The data set has five rows.
- The data set has five columns.
- The `prop` column is of type `numeric`.
- The column `sex` is of type `factor`.